Microsoft

# Troubleshooting with the Windows Sysinternals Tools

Guidance from the tools' creator



MARK RUSSINOVICH | AARON MARGOSIS

Microsoft

# Troubleshooting with the Windows Sysinternals Tools


Mark Russinovich
Aaron Margosis

PUBLISHED BY
Microsoft Press
A division of Microsoft Corporation
One Microsoft Way
Redmond, Washington 98052-6399



Library of Congress Control Number: 2014951871
ISBN: 978-0-7356-8444-7

Printed and bound in the United States of America.

First Printing

Microsoft Press books are available through booksellers and distributors worldwide. If you need support related to this book, email Microsoft Press Support at mspinput@microsoft.com. Please tell us what you think of this book at http://aka.ms/tellpress.



**Acquisitions Editor:** Devon Musgrave
**Developmental Editor:** Carol Dillingham
**Project Editor:** Carol Dillingham
**Editorial Production:** Waypoint Press
**Technical Reviewer:** Christophe Nasarre; Technical review services provided by Content Master, a member of CM Group, Ltd.
**Copyeditor:** Roger LeBlanc
**Indexer:** Christina Palaia
**Cover:** Twist Creative • Seattle

# Contents at a glance

*This page intentionally left blank*

# Table of Contents

## Chapter 14 Network and communication utilities 423



## Chapter 15 System information utilities 437

# Foreword

The arrival of a new edition of *Troubleshooting with the Windows Sysinternals Tools* is always a treat, and when mine arrived at my country estate in Scotland, I prepared myself for a ride as exciting as my first time flying. Now, I understand that, to non-magical people (we call them Sysintuggles), it appears, against all comprehension, that the authors were trying to solve the problem of "why don't people read instruction manuals more often?" and stumbled across the baffling conclusion of "because those pamphlets are simply too small." (And they have overachieved on solving that problem, producing a volume large enough to defend against even the most vicious lycanthrope.) But they simply don't understand the magic that this work unlocks.

I settled in to have a read. Upon stroking the spine of this book, it opened placidly and I began to flip through it. This is a spell book of the highest quality, designed with practical magic in mind. Paired with the theory in Windows Internals, you'll be equipped with the finest magical education available today. Using the potions and incantations included herein, it's possible to do truly remarkable things. It can teach you to bewitch Windows and ensnare malware. It can tell you how to bottle insight, brew troubleshooting glory, and even put a stopper in bluescreens. I started annotating my book, dog-earing it, and writing related spells in the margins, and soon I had an indispensable resource. It has an honored spot on my bookshelf.

This is a powerful resource for doing truly advanced magic. If you are responsible for system administration anywhere, large or small, you have something to learn from this book. Professor Russinovich truly is the brightest wizard of his age, and he and his house-elf have created an indispensable work.

A Noted Person
May 2016

# Introduction

The Sysinternals Suite is a set of over 70 advanced diagnostic and troubleshooting utilities for the Microsoft Windows platform written by me—Mark Russinovich—and Bryce Cogswell. Since Microsoft's acquisition of Sysinternals in 2006, these utilities have been available for free download from Microsoft's Windows Sysinternals website (part of Microsoft TechNet).

The goal of this book is to familiarize you with the Sysinternals utilities and help you understand how to use them to their fullest. The book will also show you examples of how I and other Sysinternals users have leveraged the utilities to solve real problems on Windows systems.

Although I coauthored this book with Aaron Margosis, the book is written as if I am speaking. This is not at all a comment on Aaron's contribution to the book; without his hard work, this book would not exist.

> **Note** See the "Late-breaking changes" section later in this chapter for updates that occurred as we were going to publish.

## Tools the book covers

This book describes all of the Sysinternals utilities that are available on the Windows Sysinternals website (*http://technet.microsoft.com/en-us/sysinternals/default.aspx*) and all of their features as of the time of this writing (early summer, 2016). However, Sysinternals is highly dynamic: existing utilities regularly gain new capabilities, and new utilities are introduced from time to time. (To keep up, follow the RSS feed of the "Sysinternals Site Discussion" blog: *http://blogs.technet.microsoft.com/sysinternals/*.) So, by the time you read this book, some parts of it might already be out of date. That said, you should always keep the Sysinternals utilities updated to take advantage of new features and bug fixes.

This book does not cover Sysinternals utilities that have been deprecated and are no longer available on the Sysinternals site. If you are still using RegMon (Registry Monitor) or FileMon (File Monitor), you should replace them with Process Monitor, described in Chapter 5. Rootkit Revealer, one of the computer industry's first rootkit detectors (and the tool that discovered the "Sony rootkit"), has served its purpose and has been

retired. Similarly, a few other utilities (such as Newsid and EfsDump) that used to provide unique value have been retired because either they were no longer needed or `equivalent functionality was eventually added to Windows.

## The history of Sysinternals

The first Sysinternals utility I wrote, Ctrl2cap, was born of necessity. Before I started using Windows NT in 1995, I mostly used UNIX systems, which have keyboards that place the Ctrl key where the Caps Lock key is on standard PC keyboards. Rather than adapt to the new layout, I set out to learn about Windows NT device driver development and to write a driver that converts Caps Lock key presses into Ctrl key presses as they make their way from the keyboard into the Windows NT input system. Ctrl2cap is still posted on the Sysinternals site today, and I still use it on all my systems.

Ctrl2cap was the first of many tools I wrote to learn about the way Windows NT works under the hood while at the same time providing some useful functionality. The next tool I wrote, NTFSDOS, I developed with Bryce Cogswell. I had met Bryce in graduate school at Carnegie Mellon University, and we had written several academic papers together and worked on a startup project where we developed software for Windows 3.1. I pitched the idea of a tool that would allow users to retrieve data from an NTFS-formatted partition by using the ubiquitous DOS floppy. Bryce thought it would be a fun programming challenge, and we divided up the work and released the first version about a month later.

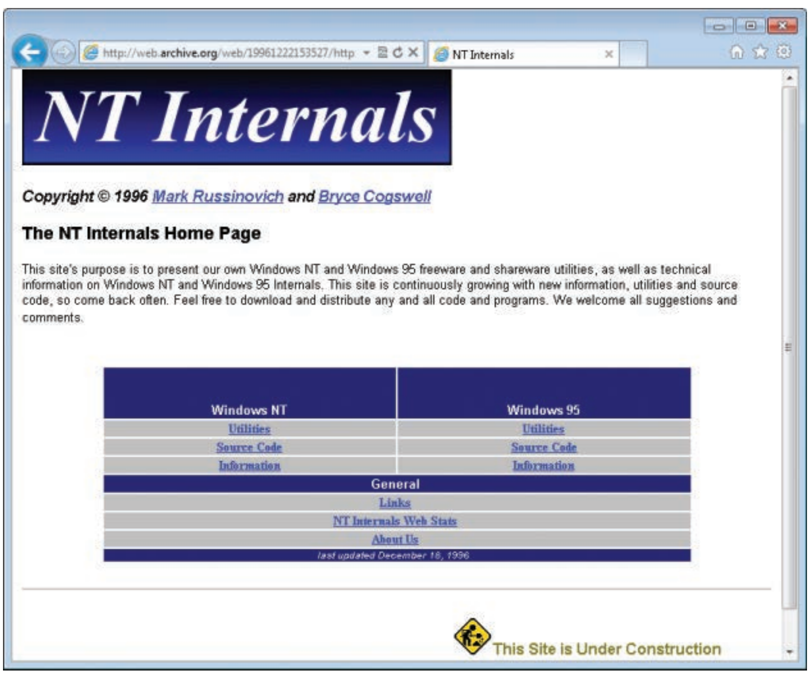
I also wrote the next two tools, Filemon and Regmon, with Bryce. These three utilities—NTFSDOS, Filemon, and Regmon—became the foundation for Sysinternals. Filemon and Regmon, both of which we released for Windows 95 and Windows NT, showed file system and registry activity, becoming the first tools anywhere to do so and making them indispensable troubleshooting aids.

Bryce and I decided to make the tools available for others to use, but we didn't have a website of our own, so we initially published them on the site of a friend, Andrew Schulman, who I'd met in conjunction with his own work uncovering the internal operation of DOS and Windows 95. Going through an intermediary didn't allow us to update the tools with enhancements and bug fixes as quickly as we wanted, so in September 1996 Bryce and I created NTInternals.com to host the tools and articles we wrote about the internal operation of Windows 95 and Windows NT. Bryce and I had also developed tools that we decided we could sell for some side income, so in the same month, we also founded Winternals Software, a commercial software company that we bootstrapped by driving traffic with a single banner ad on NTInternals.com. The first

utility we released as Winternals Software was NTRecover, a utility that enabled users to mount the disks of unbootable Windows NT systems from a working system and access them as if they were locally attached disks.

The mission of NTInternals.com was to distribute freeware tools that leveraged our deep understanding of the Windows operating system in order to deliver powerful diagnostic, monitoring, and management capabilities. Within a few months, the site, shown in the following screenshot as it looked in December 1996 (thanks to the Internet Archive's Wayback Machine), drew 1,500 visitors per day, making it one of the most popular utility sites for Windows in the early days of the internet revolution. In 1998, at the "encouragement" of Microsoft lawyers, we changed the site's name to Sysinternals.com.

Over the next several years, the utilities continued to evolve. We added more utilities as we needed them, as our early power users suggested enhancements, or when we thought of a new way to show information about Windows.




The Sysinternals utilities fell into three basic categories: those used to help programmers, those for system troubleshooting, and those for systems management. DebugView, a utility that captures and displays program debug statements, was one of the early developer-oriented tools that I wrote to aid my own development of

device drivers. DLLView, a tool for displaying the DLLs that processes have loaded, and HandleEx, a process-listing GUI utility that showed open handles, were two of the early troubleshooting tools. (I merged DLLView and HandleEx to create Process Explorer in 2001.) The PsTools, discussed in Chapter 7, are some of the most popular management utilities, bundled into a suite for easy download. PsList, the first PsTool, was inspired initially by the UNIX *ps* command, which provides a process listing. The utilities grew in number and functionality, becoming a software suite of utilities that allowed you to easily perform many tasks on a remote system without requiring installation of special software on the remote system beforehand.

Also in 1996, I began writing for Windows IT Pro magazine, highlighting Windows internals and the Sysinternals utilities and contributing additional feature articles, including a controversial article in 1996 that established my name within Microsoft itself, though not necessarily in a positive way. The article, "Inside the Difference Between Windows NT Workstation and Windows NT Server," pointed out the limited differences between Windows NT Workstation and Windows NT Server, which contradicted Microsoft's marketing message.

I exacerbated Microsoft's negative view of me by releasing Ntcrash and Ntcrash2, tools that are now called "fuzzers," that barraged the Windows NT system call interface with random garbage. The tools identified several dozen system calls that had weak parameter validation that allowed memory corruption and blue-screen crashes by unprivileged user-mode processes. (In the threat landscape of the 1990s, these were simply considered reliability bugs and were embarrassing—today they'd be classified as "important" security bugs.)

As the utilities continued to evolve and grow, I began to contemplate writing a book on Windows internals. Such a book already existed, *Inside Windows NT* (Microsoft Press, 1992), the first edition of which was written by Helen Custer alongside the original release of Windows NT 3.1. The second edition was rewritten and enhanced for Windows NT 4.0 by David Solomon, a well-established operating system expert, trainer, and writer who had worked at DEC. Instead of writing a book from scratch, I contacted him and suggested that I coauthor the third edition, which would cover Windows 2000. My relationship with Microsoft had been on the mend since the 1996 article as the result of my sending Windows bug reports directly to Windows developers, but David still had to obtain permission, which Microsoft granted.

As a result, David Solomon and I coauthored the third, fourth, fifth, and sixth editions of the book, which we renamed *Windows Internals* at the fourth edition. We brought in Alex Ionescu as a co-author beginning with the fifth edition. By the sixth edition, the content had grown so much that we needed to split the book into two parts. Not long after we finished *Inside Windows 2000* (Microsoft Press, 2000), I joined

David to teach his Windows internals seminars, adding my own content. Offered around the world, even at Microsoft to the developers of Windows, these classes have long used the Sysinternals utilities to show students how to peer deep into Windows internals and learn more when they returned to their developer and IT professional roles at home.

By 2006, my relationship with Microsoft had been strong for several years, Winternals had a full line of enterprise management software and had grown to about 100 employees, and Sysinternals had two million downloads per month. On July 18, 2006, Microsoft acquired Winternals and Sysinternals. Not long after, Bryce and I (there we are below in 2006) moved to Redmond to become a part of the Windows team. Today, I serve as the Chief Technology Officer of Microsoft Azure, leading the technical strategy and architecture of the Azure cloud computing platform.



Two goals of the acquisition were to make sure that the tools Bryce and I developed would continue to be freely available and that the community we built would thrive, and they have. Today, the Windows Sysinternals site on technet.microsoft.com is one of the most frequently visited sites on TechNet, averaging 4.5 million downloads per month. Sysinternals power users come back time and again for the latest versions of the utilities and for new utilities, such as the recently released Sysmon and PsPing, as well as to participate in the Sysinternals community, a growing forum with over 42,000 registered users at the time of this writing. I remain dedicated to continuing to enhance the existing tools and to add new tools.

Many people suggested that a book on the tools would be valuable, but it wasn't until David Solomon suggested that one was way overdue that I started the project. My responsibilities at Microsoft did not permit me to devote the time necessary to write another book, but David pointed out that I could find someone to help. I was pleased that Aaron Margosis agreed to partner with me. Aaron is a Principal Consultant with Microsoft Cybersecurity Services who is known for his deep understanding of Windows security and application compatibility. I have known Aaron for many years, and his

excellent writing skills, familiarity with Windows internals, and proficiency with the Sysinternals tools made him an ideal coauthor.

## Who should read this book

This book exists for Windows IT professionals, power users, and even developers who want to make the most of the Sysinternals tools. Regardless of your experience with the tools, and whether you manage the systems of a large enterprise, a small business, or the PCs of your family and friends, you're sure to discover new tools, pick up tips, and learn techniques that will help you more effectively troubleshoot the toughest Windows problems and simplify your system-management operations and monitoring.

### Assumptions

This book expects that you have familiarity with the Windows operating system. Basic familiarity with concepts such as processes, threads, virtual memory, and the Windows command prompt is helpful, though some of these concepts are discussed in Chapter 2, "Windows core concepts."

## Organization of this book

The book is divided into three parts. Part I, "Getting started," provides an overview of the Sysinternals utilities and the Sysinternals website, describes features common to all of the utilities, tells you where to go for help, and discusses some Windows core concepts that will help you better understand the platform and the information reported by the utilities.

Part II, "Usage guide," is a detailed reference guide covering all of the Sysinternals utilities' features, command-line options, system requirements, and caveats. With plentiful screenshots and usage examples, this section should answer just about any question you have about the utilities. Major utilities such as Process Explorer and Process Monitor each get their own chapter; subsequent chapters cover utilities by category, such as security utilities, Active Directory utilities, and file utilities.

Part III, "Troubleshooting—'The Case of the Unexplained…'," contains stories of real-world problem solving using the Sysinternals utilities from Aaron and me, as well as from administrators and power users from around the world.

## Conventions and features in this book

This book presents information using conventions designed to make the information readable and easy to follow:

- Boxed elements with labels such as "Note" provide additional information or alternative methods for completing a step successfully.
- Text that you type (apart from code blocks) appears in bold.
- A plus sign (+) between two key names means that you must press those keys at the same time. For example, "Press Alt+Tab" means that you hold down the Alt key while you press the Tab key.
- A vertical bar between two or more menu items (for example, File | Close), means that you should select the first menu or menu item, and then the next, and so on.
- In command-line syntax specifications, a vertical bar means "OR," square braces mean "optional," italicized text is a placeholder for information that you provide, curly braces represent groupings, and ellipses represent a repeating pattern. Consider this example:

```
procdump
    [-ma | -mp | -d callback_DLL] [-64] [-r [1..5] [-a]] [-o]
    [-n count] [-s secs]
    [-c|-cl percent [-u]] [-m|-ml commit] [-p|-pl counter_threshold]
    [-e [1 [-g] [-b]]] [-h] [-l] [-t] [-f filter,...]
    {
      {{[-w] process_name}|service_name|PID } [dump_file | dump_folder] } |
      {-x dump_folder image_file [arguments]}
    }
```

This indicates that you can optionally use **–ma**, **–mp**, or **–d**; if you use **–d**, you must supply a value for *callback_DLL*. You can also choose to use the **–f** option; if you do, you must supply one or more *filter* values. The groupings in the last four lines show that you must specify a *process_name*, *service_name*, or *PID*, or use the **–x** option with a *dump_folder* and *image_file*.

## System requirements

The Sysinternals tools work on the following supported versions of Windows, including 64-bit editions, unless otherwise specified:

- Windows Vista
- Windows 7
- Windows 8.1
- Windows 10 (desktop)[1]
- Windows Server 2008
- Windows Server 2008 R2
- Windows Server 2012
- Windows Server 2012 R2
- Windows Server 2016, including Nano Server

Some tools require administrative rights to run, and others implement specific features that require administrative rights.

## Late-breaking changes

Just as we were finishing work on this book, I released updated versions of many of the utilities to support the Nano Server edition of Windows Server 2016. Nano Server is a small-footprint, headless installation option for Windows Server 2016 that includes a minimal number of features and services. Of particular interest to Sysinternals users is that Nano Server does not include a 32-bit subsystem nor GUI components. As described in Chapter 1, "Getting started with the Sysinternals utilities," each Sysinternals utility has always been packaged as a single 32-bit executable, with any additional required files, such as 64-bit binaries, embedded as resources that can be extracted and executed as needed. Of course, none of these 32-bit images would work on Nano Server, so I created native 64-bit versions of the console-mode utilities, appending "64.exe" to their file names. For example, the 64-bit version of SigCheck.exe is

[1] The Sysinternals utilities are all Win32 apps, support only x86 and x64 architectures and are not compatible with Windows 10 Mobile, IoT, Xbox, etc.

SigCheck64.exe. In addition, I created a console-mode version of the LoadOrd (Load Order) utility, LoadOrdC.exe, and a native 64-bit version, LoadOrdC64.exe.

Nano Server management relies heavily on PowerShell Remoting. PowerShell treats any output to the standard error (stderr) stream as indicative of an error. The console-mode Sysinternals utilities had always written banner and syntax information to stderr. To improve the utilities' support for PowerShell and for Nano Server in particular, the utilities now write banner and syntax information to the standard output (stdout) stream, and use the new **–nobanner** command-line option to omit banner output. Note that this replaces the **–q** option that many of the utilities had used for the same purpose.

## Acknowledgments

First, Aaron and I would like to thank Bryce Cogswell, cofounder of Sysinternals, for his enormous contribution to the Sysinternals tools. Because of our great collaboration, what Bryce and I published on Sysinternals was more than just the sum of our individual efforts. Bryce retired from Microsoft in October 2010, and we wish him luck in whatever he pursues.

We'd like to thank David Solomon for spurring Mark to write this book, providing detailed review of many chapters, and writing the Foreword for the first edition. Dave has also been one of Sysinternals most effective evangelists over the years and has suggested many valuable features.

Thanks to Luke Kim, who has been invaluable in helping upgrade the projects to the latest versions of Microsoft Visual Studio, moving the tools into Visual Studio Team Services (VSTS) source control, streamlining the build and publishing process, and managing the Sysinternals.com website and live.sysinternals.com infrastructure servers (which are running on Azure). Thanks also to Kent Sharkey for publishing updates to Sysinternals.com.

Up until a few years ago, Bryce and I were the sole authors of the tools, but I started accepting contributions from other developers. Ken Johnson, Andrew Richards, Thomas Garnier, David Magnotti, Dmitry Davydok, Daniel Pearson, Justin Jiang and the rest of the Nano Server team, Giulia Biagini, Pavel Yosifovich, and Aaron Margosis have all added significant features to specific tools.

Huge thanks to John Sheehan for his help describing previously-undocumented details about how AppContainers work; to Alex Ionescu for material relating to

protected processes; and to Ned Pyle, Marty Lichtel, and Carl Harrison for allowing us to incorporate cases they had previously published.

We are grateful to the following people who provided valuable and insightful technical review, corrections, and suggestions for this edition of the book: Andrew Richards, Bhaskar Rastogi, Bruno Aleixo, Burt Harris, Chris Jackson, Crispin Cowan, Greg Cottingham, Ken Johnson (a.k.a., Skywing), Luke Kim, Mario Raccagni, Steve Thomas, and Yong Rhee.

Aaron and I considered it a longshot when we asked Noted Person to consider writing the Foreword for this edition, and we are still giddy and starstruck that Noted Person agreed. Our unbounded thanks to N.P.[2]

We'd like to thank Devon Musgrave (acquisitions editor and developmental editor) and Carol Dillingham (project editor) from Microsoft Press for all the great work they have done for us on this edition, and especially for their infinite patience as we slipped our deadline from a fixed date to something closer to "infinity." Thanks to Steve Sagman from Waypoint Press for project management and desktop publishing. Thanks also to Christophe Nasarre for technical editing and Roger LeBlanc for copyediting.

Aaron thanks his wife, Elise, and their children—Elana, Jonah, and Gabriel—for their love and support. Aaron also thanks Brenda Schrier for his author photo. Aaron also thanks the Washington Nationals Baseball Club and West Ham United F.C.

Mark thanks his wife, Daryl, and daughter, Maria, for supporting all his endeavors.

## Errata, updates, and book support

We've made every effort to ensure the accuracy of this book. You can access updates to this book—in the form of a list of submitted errata and their related corrections—at:

*http://aka.ms/TroubleshootSysint/errata*

If you find an error that is not already listed, you can report it to us through the same page.

If you need additional support, e-mail Microsoft Press Book Support at *mspinput@microsoft.com*.

[2] Noted Person's secret identity is *Chris Jackson*, a.k.a., The App Compat Guy, a.k.a., Captain Inappropriate.

Please note that product support for Microsoft software is not offered through the previous addresses. For help with Microsoft software or hardware, go to *http://support.microsoft.com*.

## Free ebooks from Microsoft Press

From technical overviews to in-depth information on special topics, the free ebooks from Microsoft Press cover a wide range of topics. These ebooks are available in PDF, EPUB, and Mobi for Kindle formats, ready for you to download at:

*http://aka.ms/mspressfree*

Check back often to see what is new!

## We want to hear from you

At Microsoft Press, your satisfaction is our top priority, and your feedback our most valuable asset. Please tell us what you think of this book at:

*http://aka.ms/tellpress*

The survey is short, and we read every one of your comments and ideas. Thanks in advance for your input!

## Stay in touch

Let's keep the conversation going. Follow Microsoft Press on Twitter: *http://twitter.com/MicrosoftPress*.

*This page intentionally left blank*

PART I

# Getting started

CHAPTER 4

# Autoruns

A question I often hear is, "Why is all this *stuff* running on my computer?" That's often followed with, "How do I get rid of it?" The Microsoft Windows operating system is a highly extensible platform. Not only can programmers write applications that users can choose to run, those programmers can "add value" by having their software run automatically without troubling the user to start it, by adding visible or nonvisible features to Windows Explorer and Internet Explorer, or by supplying device drivers that can interact with custom hardware or change the way existing hardware works. Sometimes the "value" to the user is doubtful at best; sometimes the value is for someone else entirely and the software acts to the detriment of the user (which is when the software is called *malware*).

*Autostarts* is the term I use to refer to software that runs automatically without being intentionally started by a user. This type of software includes drivers and services that start when the computer is booted; applications, utilities, and shell extensions that start when a user logs on; and browser extensions that load when Internet Explorer is started. Over 200 locations in the file system and registry allow autostarts to be configured on x64 versions of Windows. These locations are often referred to as *Autostart Extensibility Points*, or ASEPs.

ASEPs have legitimate and valuable purposes. For example, if you want your instant messaging contacts to know when you are online, having the messaging client start when you log on is a great help. Users enjoy search toolbars and PDF readers that become part of Internet Explorer. And much of Windows itself is implemented through ASEPs in the form of drivers, services, and Explorer extensions.

On the other hand, consider the plethora of "free" trial versions of programs that computer manufacturers install on new computers and that fill up the taskbar notification area. Consider also the semihidden processes that legitimate vendors run all the time so that their applications can appear to start more quickly. Do you really need all these processes constantly consuming resources? On top of that, malware almost always hooks one or more ASEPs, and virtually every ASEP in Windows has been used by malware at one point or another.

Although Windows Vista and Windows 7 offer the System Configuration Utility (msconfig.exe, shown in Figure 4-1) to let you see some of these autostarts, it shows only a small subset and is of limited use. Msconfig also requires administrative rights, even just to view settings. That means it cannot identify or disable *per-user* autostarts belonging to nonadministrator users.

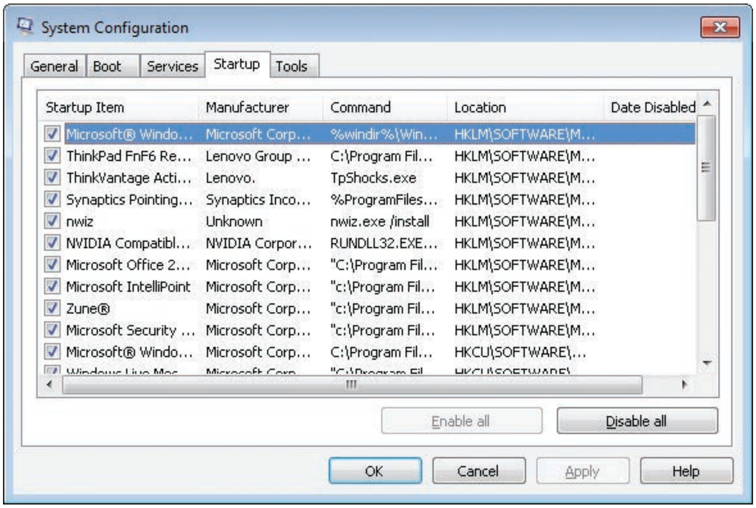



**FIGURE 4-1** The MSConfig utility included in Windows Vista and Windows 7 exposes a limited set of autostarts.

Some of MSConfig's functionality moved into Task Manager when Windows 8 introduced a Startup tab, as shown in Figure 4-2. Although it no longer requires administrative rights, it no longer shows a process' full command line, nor where the ASEP is configured.

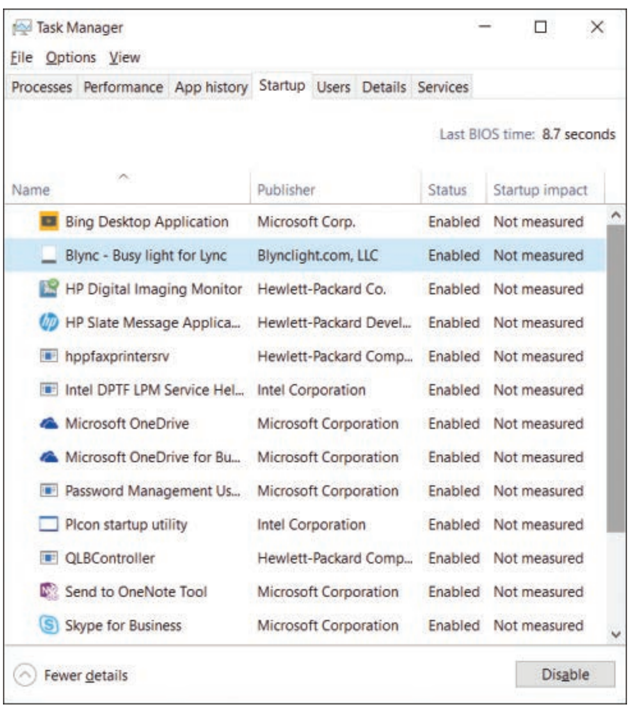



**FIGURE 4-2** The Task Manager Startup tab in Windows 8 and newer is not much of an improvement.

Bryce and I created the Autoruns utility to expose as many autostarts as we could identify, and to make it easy to disable or remove those autostarts. The information that Autoruns exposes can be discovered manually if you know where to look in the registry and file system. Autoruns automates that task, scanning a large number of ASEPs in a few seconds, verifying entries, and making it easier to identify entries with suspicious characteristics, such as the lack of a digital signature, or that are flagged as suspicious by VirusTotal. We also created a command-line version, AutorunsC, to make it possible to capture the same information in a scripted fashion.

Using either Autoruns or AutorunsC, you can easily capture a baseline of the ASEPs on a system. That baseline can be compared against results captured at a later time so that changes can be identified for troubleshooting purposes. Many organizations use Autoruns as part of a robust change-management system, capturing a new baseline whenever the desktop image is updated.

## Autoruns fundamentals

Launch Autoruns and it immediately begins filling its display with entries collected from known ASEPs. As shown in Figure 4-3, each shaded row represents an ASEP location, with a Regedit icon if it is a registry location or a folder icon if it is stored in the file system.[1] The rows underneath a shaded row indicate entries configured in that ASEP. Each row includes the name of the autostart entry; the description, publisher and timestamp of the item; and the path to the file to run and an icon for that file. Each row also has a check box to temporarily disable the entry, and a column to display VirusTotal results. A panel at the bottom of the window displays details about the selected entry, including its full command line. The Everything tab, which is displayed when Autoruns starts, displays all ASEP entries on the system; you use the 19 other tabs to view just specific categories of autostarts. Each of these categories will be described later in this chapter.

The Image Path column shows the full path to the target file identified by the autostart entry. In some cases, this will be the first name in the autostart's command line. For autostarts that use a hosting process—such as Cmd.exe, Wscript.exe, Rundll32.exe, Regsvr32.exe, or Svchost.exe—the image path identifies the target script or DLL on the command line instead of the main executable. For entries that involve levels of indirection, Autoruns follows the indirection to identify the target image. For example, the Internet Explorer "Browser Helper Objects" ASEPs are recorded as GUIDs in the registry; Autoruns identifies the corresponding InProcServer entries under HKCR\CLSID and reports those DLLs. If the target file cannot be found in the expected location, the Image Path column will include the text "File not found" and the entry will be highlighted in yellow.

1 Scheduled Tasks appear with a folder icon, because configuration settings for tasks were stored in %windir%\Tasks prior to Windows Vista. As part of the re-architecting of Task Scheduler, configuration settings are now in the registry under HKLM\Software\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache and in the file system under System32\Tasks.

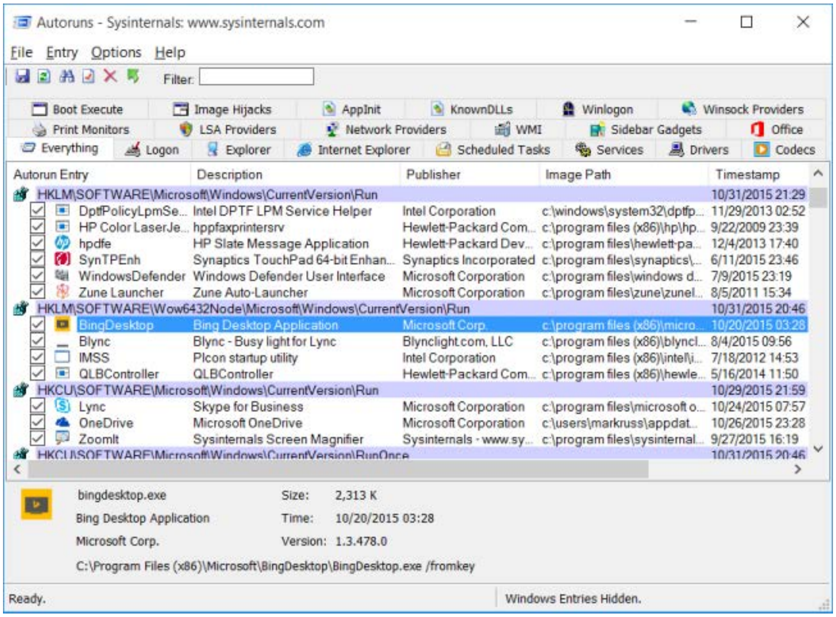



**FIGURE 4-3** Autoruns main window.

If the file identified in the image path is a Portable Executable (PE) file, the Timestamp column reports the date and time in the local time zone in which the image was created by the linker; otherwise—for example, for script files—the timestamp reports the last-write time of the file according to the file system. For the shaded rows that identify an ASEP location, the timestamp reports the last-modification time for the registry key or file system directory.

The Description and Publisher columns in the display are taken from the Description and Company Name version resources, respectively, for files that contain version resources, such as EXE and DLL files. If the file's digital signature has been verified, the Publisher column displays the subject name from the corresponding code-signing certificate. (See the "Verifying code signatures" section later in this chapter for more information.)

The Description and Publisher columns are left blank if the target file cannot be found, has no Description and Company Name in its version resources, or has no version resource (which is always true of script files). The VirusTotal column is blank until you request information from the VirusTotal service, as described in the "VirusTotal analysis" section later in this chapter.

Autoruns calls attention to suspicious images by highlighting their entries in pink. Autoruns considers an image file suspicious if it has no description or publisher, or if signature verification is enabled and the image doesn't have a valid signature.

You can quickly search for an item by pressing Ctrl+F and entering text to search for. Autoruns will select the next row that contains the search text. Pressing F3 repeats the search from the current location. Pressing Ctrl+C copies the text of the selected row to the clipboard as tab-delimited text.

On the Options menu, the Scan Options entry is disabled while Autoruns is scanning the system. To cancel the scan so that you can change those options (which are described later in this chapter), press the Esc key. A change to any selection in Scan Options takes effect during the next scan. To run a new scan with the same options, press F5 or click the Refresh button on the toolbar.

## Disabling or deleting autostart entries

With Autoruns, you can disable or delete autostart entries. Deleting an entry permanently removes it, and you should do this only if you're certain you never want the software to autostart again. Select the entry in the list, and press the Del key. Because there is no Undo, Autoruns prompts for confirmation before deleting the autostart entry.

By contrast, when you disable an entry by clearing its check box, Autoruns leaves a marker behind that Autoruns recognizes and with which it can reconstitute and re-enable the entry. For example, for most registry ASEPs, Autoruns creates an *AutorunsDisabled* subkey in the ASEP location and copies the registry value being disabled into that subkey before deleting the original value. Windows will not process anything in that subkey, so the items in it will not run, but Autoruns displays them as disabled autostarts. Checking the entry again puts the entry back into the actual ASEP location. For ASEPs in the file system such as in the Start menu, Autoruns creates a hidden directory named AutorunsDisabled and moves disabled entries into that directory.

Note that disabling or deleting an autostart entry prevents it from being automatically started in the future. It does not stop any existing processes, nor does it delete or uninstall the ASEP's target file.

Also note that if you disable autostarts that are critical for system boot, initialization, or correct operation, you can put the system into a state in which recovery is not possible without booting into an alternate operating system or recovery environment.

## Autoruns and administrative permissions

The vast majority of ASEPs are in locations that grant Read permission to standard users. On some versions of Windows, the registry keys containing configuration information for some services are locked down, and many scheduled tasks are not standard-user readable. But for the most part, Autoruns works perfectly fine without administrative rights for the purposes of viewing autostart entries.

Administrative rights are required to view *all* autostarts, and they are required if you need to change the state of entries in systemwide locations, such as HKLM or the all users' Startup directory in the Start menu. If you select or clear a check box, or try to delete one of these entries without administrative rights, Autoruns will report Access Denied. The error message dialog box includes a *Run As Administrator* button that lets you restart Autoruns elevated. (See Figure 4-4.) When Autoruns has administrative rights, configuration changes should succeed. You can also restart Autoruns with User Account Control (UAC) elevation by choosing Run As Administrator from the File menu.

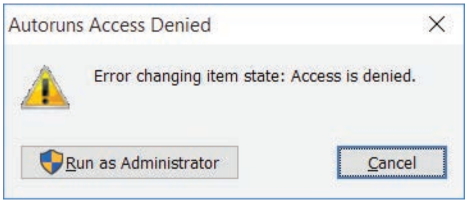



**FIGURE 4-4** Access Denied and the option to restart Autoruns with UAC elevation.

To ensure that Autoruns has elevated rights when it launches, start Autoruns with the **–e** command-line option. This will request UAC elevation if the invoker is not already running elevated. See the "Administrative rights" section in Chapter 2, "Windows core concepts," for more information on UAC elevation.

## Verifying code signatures

Anyone can create a program and stick the name "Microsoft Corporation" in it. Therefore, seeing that text in the Publisher column gives only a low degree of assurance that the file in question was created by Microsoft and has not been modified since. Verifying a digital signature associated with that file gives a much higher degree of assurance of the file's authenticity and integrity. The file format for some types of files allows for a digital signature to be embedded within the file. Files can also be *catalog-signed*, meaning that the information needed to validate a file's content is in a separate file. Catalog signing means that even plain text files can be verified.

You can verify an entry's digital signature by selecting the entry and choosing Verify Image from the Entry menu. If the file has been signed with a valid code-signing certificate that derived from a root certificate authority that is trusted on the computer, the text in the Publisher column changes to "(Verified)" followed by the subject name in the code-signing certificate. If the file has not been signed or the verification fails for any other reason, the text changes to "(Not verified)" followed by the company name from the file's version resource, if present.

Instead of verifying entries one at a time, you can enable Verify Code Signatures in the Scan Options dialog box and rescan. Autoruns will then attempt to verify the signatures for all image paths as it scans autostarts. Note that the scan might take longer because it also verifies whether each signing certificate has been revoked by its issuer, which requires Internet connectivity to work reliably.

Files for which signature checks fail might be considered suspicious and therefore appear in pink. A common malware technique is to install files that on casual inspection appear to be legitimate Windows files but are not signed by Microsoft.

The Sysinternals SigCheck utility, described in Chapter 9, "Security utilities," provides deeper detail for file signatures, including whether the file is catalog-signed and the location of the catalog.

## VirusTotal analysis

VirusTotal.com is a free web service that lets users upload files to be analyzed by over 50 antivirus engines and see the results of those scans. Most users interact with VirusTotal by opening a web browser to *https://www.virustotal.com* and uploading one file at a time. VirusTotal also offers an API for programs such as Autoruns that makes it possible not only to scan many files at once, but also to do so much more efficiently by uploading only file hashes rather than entire files. If VirusTotal has recently received a file with the same hash, it returns the results from the most recent scan rather than performing the scan again.

You can analyze all autostart entries by enabling Check VirusTotal.com in the Scan Options dialog box and rescanning. Autoruns uploads file hashes to VirusTotal.com and writes "Hash submitted…" in the VirusTotal column. As results come back, Autoruns replaces the text in that column with the number of engines that flagged the file out of the total number of engines that returned results, rendered as a hyperlink, as shown in Figure 4-5. As an additional visual indicator, the link is colored red if any engines flagged the file as suspicious. Click the link to open the webpage where you can see details of the results. If VirusTotal has no record of the file's hash, Autoruns reports "Unknown."

If you also enable Submit Unknown Images in the Scan Options dialog box, Autoruns automatically uploads the entire file to VirusTotal in response to an "Unknown" report. Uploading and scanning complete files can take several minutes, during which time Autoruns displays a "Scanning…" hyperlink in the VirusTotal column. Click that link to view the progress of the analysis.

You can also analyze items one at a time by right-clicking an autostart and choosing Check VirusTotal from the popup menu. Autoruns sends the file's hash to VirusTotal and reports the engines' results for that entry or "Unknown." You can then upload the full file by right-clicking the entry again and choosing Submit To VirusTotal (if it was unknown) or Resubmit To VirusTotal (to force a new scan).

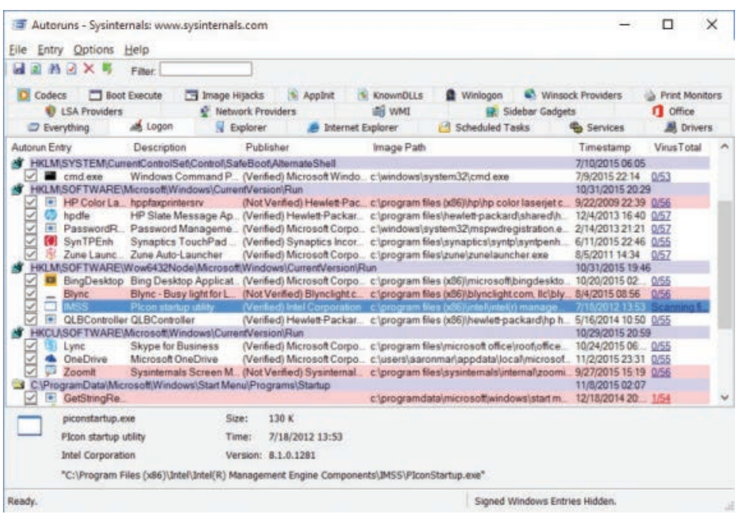



**FIGURE 4-5** Autoruns with VirusTotal results.

You have to agree to VirusTotal's terms of service before using the Sysinternals utilities to query VirusTotal. On first use of VirusTotal, Autoruns will open your default web browser to the VirusTotal terms of service page and prompt you in a message box to agree with the terms before proceeding.

See Chapter 3, "Process Explorer," for additional considerations regarding VirusTotal analysis, and in particular regarding uploading files to the VirusTotal service.

## Hiding entries

The default list of ASEP entries is always large because, as mentioned earlier, Windows itself makes extensive use of ASEPs. Typically, Windows' own autostart entries are not of interest when troubleshooting. Likewise, autostart entries from other Microsoft-published software such as Microsoft Office are usually not the droids you're looking for[2]. And when enabling VirusTotal analysis, you're probably more interested in inspecting the non-zero results than the entries that no antimalware engine has marked.

Autoruns offers several choices on the Options menu to show only those more-interesting entries, and a "filter" feature on the Autoruns toolbar to show only items containing the text you specify. None of these options requires rescanning the system; they manipulate the previously-collected results and can show hidden entries again instantly on demand.

You can choose to hide Windows and Microsoft autostart entries from the display by enabling the Hide Windows Entries or Hide Microsoft Entries from the Options menu. The Hide Windows Entries option is enabled by default. Enabling Hide Microsoft Entries also enables Hide Windows Entries. If the entry is a hosting process such as Cmd.exe or Rundll32.exe, the filter options' logic is based on whether the target file is a Windows or Microsoft image and whether it is signed.

The behavior of these two options depends on whether Verify Code Signatures is also enabled. If signature verification is not enabled, Hide Windows Entries omits from the display all entries for which the target image file has the word "Microsoft" in the version resource's Company Name field, and for which the image file resides in or below the %windir% directory. Hide Microsoft Entries checks only for "Microsoft" in the Company Name field and omits those entries. As mentioned earlier, it is easy for anyone to create a program that gets past this check, so the Verify Code Signatures option is highly recommended.

If signature verification is enabled, Hide Windows Entries omits entries that are signed with the Microsoft Windows code-signing certificate. (Windows components are signed with a different cer-

2 Cultural reference: "These aren't the droids you're looking for" is a quote from the film, *Star Wars IV: A New Hope*.

tificate from other Microsoft products.) Hide Microsoft Entries omits entries that are signed with any Microsoft code-signing certificate that chains to a trusted root certificate authority on the computer.

**Note** Some files that ship with Windows, particularly drivers, are provided by third parties and have a third-party name in the Company Name field of the file's version resource, but they are catalog signed with the Windows code-signing certificate. Consequently, these entries can be hidden when signature verification is enabled but displayed when verification is not enabled. The SigCheck utility described in Chapter 9 reports both the Company Name and the name from the signing certificate. The AutorunsC utility described later in this chapter can report both also.

If you enable Hide VirusTotal Clean Entries in the Options menu, Autoruns removes from the display all entries for which VirusTotal reports zero issues. Autoruns shows only entries that are flagged by one or more VirusTotal engines, that are unknown to VirusTotal, or that couldn't be queried because the file couldn't be found or was inaccessible to Autoruns. On a typical system, this option should hide most entries. Note that when a small number of the VirusTotal engines report an issue, it is usually a false positive.

Another great way to find items of interest is to type search text in the Filter text entry field in the toolbar, as shown in Figure 4-6. As you type, Autoruns limits the displayed entries to rows that contain the exact (case-insensitive) text that you type. To remove the filter, simply delete the text from the entry field.

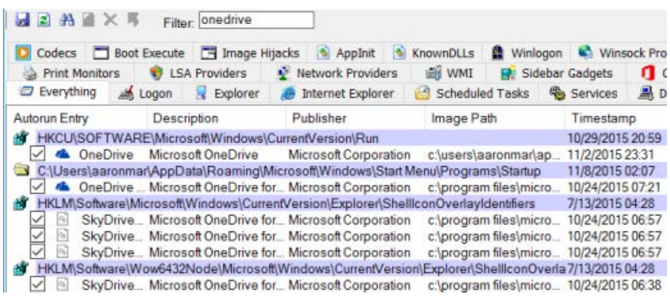



**FIGURE 4-6** The Filter text box limits Autoruns results to entries containing the word "onedrive."

By default, Autoruns displays a shaded row only for ASEPs that have entries configured within them and that are not hidden. If Hide Empty Locations is disabled in the Options menu, Autoruns displays a shaded row for every ASEP that it scans, whether it has entries or not. Autoruns scans a tremendous number of ASEPs, so this increases the amount of output dramatically. Disabling this option can be useful to verify whether particular ASEPs are scanned, or to satisfy curiosity.

Scan and filter selections from the Options menu are displayed in the status bar and are saved in the registry. They'll remain in effect the next time the same user starts Autoruns.

## Getting more information about an entry

Right-clicking an entry displays the Entry submenu as a popup context menu. Five of those menu items use other programs to display more information about the selected entry than is displayed in Autoruns:

- **Jump To Entry** Opens the location where the autostart entry is configured. For ASEPs configured in the registry, Jump To Entry starts the registry editor (Regedit.exe) and sends it simulated keystrokes to navigate to the autostart entry. (If Regedit does not navigate to the correct location the first time, try the Jump To Entry command again.) For ASEPs configured in the file system, Jump To Entry opens a new Windows Explorer folder window in that location. For Scheduled Tasks, Jump To opens the Task Scheduler user interface; however, it does not try to navigate to the selected task. Note that Autoruns' driving of the navigation of Regedit requires that Autoruns not be running at a lower integrity level than Regedit.
- **Jump To Image** Opens a new Windows Explorer folder window with the file identified as the target image selected.
- **Process Explorer** If the image path is an executable (as opposed to a script or DLL file) and a process with that name is still running, Autoruns tries to get Process Explorer (Procexp) to display its Process Properties dialog box for the process. For this option to work, Procexp needs to be in the same directory with Autoruns, found in the path, or already running. If Procexp is already running, it cannot be at a higher integrity level than Autoruns. For example, if Autoruns is not elevated and Procexp is, this option will not work.
- **Search Online** Initiates an online search for the file name using your default browser and search engine.
- **Properties** Displays the Windows Explorer file Properties dialog box for the target image path.

## Viewing the autostarts of other users

If Autoruns is running with administrative rights, it adds a User item to the menu, listing the account names that have logged on to the computer and have an accessible user profile. Selecting a user account from that menu rescans the system, searching that user's ASEPs, including the Run keys under that user's HKCU and the Startup directory in that user's profile. If Show Only Per-User Locations is selected in the Scan Options dialog box, Autoruns displays only per-user ASEPs and hides all machinewide ASEPs.

One example of when this option is useful is if a standard user has installed some harmful software. With only standard user privileges, only the user's per-user ASEPs could have been modified. Software that has only standard user privileges cannot modify systemwide settings nor touch the accounts of other users on the system. Rather than logging on and allowing that malware to run—and possibly interfering with an Autoruns scan—you can log on to the system with an administrative

account, start Autoruns, select the potentially compromised account from the User menu, inspect the user's ASEPs, and perform a cleanup if problems are identified. Enabling the Scan Only Per-User Locations option makes this task even easier by hiding all the ASEPs that the non-admin user could not have configured.

## Viewing ASEPs of an offline system

Autoruns allows you to view the ASEPs of an offline instance of Windows from a different, known-good instance of Windows. This can be helpful in several scenarios:

- If Windows will not start, offline analysis can identify and remove faulty or misconfigured ASEPs.
- Malware, and rootkits in particular, can prevent Autoruns from accurately identifying ASEPs. For example, a rootkit that intercepts and modifies registry reads can hide the content of selected keys from Autoruns. By taking the system offline and viewing its ASEPs from an instance of Windows in which that malware is not running, those entries will not be hidden.
- Malicious files on your system might appear to be signed by a trusted publisher, when in fact the root certificate might also have come from the attacker. A known-good system in which the bogus certificate is not installed will fail the signature verification for those files.

To perform offline analysis, Autoruns must run with administrative rights and must have access to the offline instance's file system. Choose Analyze Offline System from the File menu, and then identify the target's Windows (System Root) directory and a user's profile directory, as shown in Figure 4-7. Autoruns then scans that instance's directories and registry hives for its ASEPs. Note that the registry hives cannot be on read-only media.

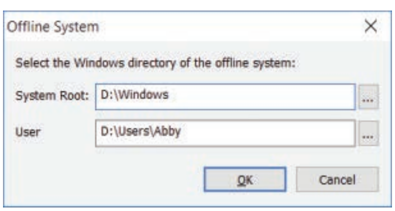



**FIGURE 4-7** Picking system and user profile directories of an offline system.

## Changing the font

Choose Font from the Options menu to change the font Autoruns uses to display its results. Changing the font updates the display immediately.

## Autostart categories

When you launch Autoruns for the first time, all autostart entries on the system are displayed in one long list on the Everything tab. As Figure 4-8 shows, the display includes up to 19 other tabs that break down the complete list into categories.

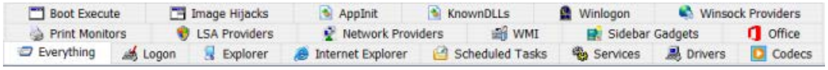



**FIGURE 4-8** Autostart categories are displayed on up to 20 different tabs.

### Logon

This tab lists the "standard" autostart entries that are processed when Windows starts up and a user logs on, and it includes the ASEPs that are probably the most commonly used by applications. They include the various Run and RunOnce keys in the registry, the Startup directories in the Start menu, computer startup and shutdown scripts, and logon and logoff scripts. It also lists the initial user session processes, such as the Userinit process and the desktop shell. These ASEPs include both per-user and systemwide locations, and entries designed for control through Group Policy. Finally, it lists the Active Setup\Installed Components keys, which although never publicly documented or supported for third-party use have been reverse-engineered and repurposed both for good and for ill.

The following lists the Logon ASEP locations that Autoruns inspects on a particular instance of an x64 version of Windows 10.

| **The Startup directory in the "all users" Start menu** |
|---|
| %ALLUSERSPROFILE%\Microsoft\Windows\Start Menu\Programs\Startup |
| **The Startup directory in the user's Start menu** |
| %APPDATA%\Microsoft\Windows\Start Menu\Programs\Startup |
| **Per-user ASEPs under HKCU\Software** |
| HKCU\Software\Microsoft\Windows\CurrentVersion\Run<br>HKCU\Software\Microsoft\Windows\CurrentVersion\RunOnce<br>HKCU\Software\Microsoft\Windows NT\CurrentVersion\Terminal Server\Install\Software\Microsoft\Windows\CurrentVersion\Run<br>HKCU\Software\Microsoft\Windows NT\CurrentVersion\Terminal Server\Install\Software\Microsoft\Windows\CurrentVersion\Runonce<br>HKCU\Software\Microsoft\Windows NT\CurrentVersion\Terminal Server\Install\Software\Microsoft\Windows\CurrentVersion\RunonceEx<br>HKCU\Software\Microsoft\Windows NT\CurrentVersion\Windows\Load<br>HKCU\Software\Microsoft\Windows NT\CurrentVersion\Windows\Run<br>HKCU\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\Shell |
| **Per-user ASEPs under HKCU\Software—64-bit only** |
| HKCU\Software\Wow6432Node\Microsoft\Windows\CurrentVersion\Run<br>HKCU\Software\Wow6432Node\Microsoft\Windows\CurrentVersion\RunOnce |

**Per-user ASEPs under HKCU\Software intended to be controlled through Group Policy**

HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer\Run
HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\System\Shell
HKCU\Software\Policies\Microsoft\Windows\System\Scripts\Logon
HKCU\Software\Policies\Microsoft\Windows\System\Scripts\Logoff

**Systemwide ASEPs in the registry**

HKLM\Software\Microsoft\Windows\CurrentVersion\Run
HKLM\Software\Microsoft\Windows\CurrentVersion\RunOnce
HKLM\Software\Microsoft\Windows\CurrentVersion\RunOnceEx
HKLM\Software\Microsoft\Active Setup\Installed Components
HKLM\Software\Microsoft\Windows NT\CurrentVersion\Terminal Server\Install\Software\Microsoft\Windows\CurrentVersion\Run
HKLM\Software\Microsoft\Windows NT\CurrentVersion\Terminal Server\Install\Software\Microsoft\Windows\CurrentVersion\Runonce
HKLM\Software\Microsoft\Windows NT\CurrentVersion\Terminal Server\Install\Software\Microsoft\Windows\CurrentVersion\RunonceEx
HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\IconServiceLib
HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\AlternateShells\AvailableShells
HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\AppSetup
HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\Shell
HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\Taskman
HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\Userinit
HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\VmApplet
HKLM\System\CurrentControlSet\Control\SafeBoot\AlternateShell
HKLM\System\CurrentControlSet\Control\Terminal Server\Wds\rdpwd\StartupPrograms
HKLM\System\CurrentControlSet\Control\Terminal Server\WinStations\RDP-Tcp\InitialProgram

**Systemwide ASEPs in the registry, intended to be controlled through Group Policy**

HKLM\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer\Run
HKLM\Software\Microsoft\Windows\CurrentVersion\Policies\System\Shell
HKLM\Software\Policies\Microsoft\Windows\System\Scripts\Logon
HKLM\Software\Policies\Microsoft\Windows\System\Scripts\Logoff
HKLM\Software\Policies\Microsoft\Windows\System\Scripts\Startup
HKLM\Software\Policies\Microsoft\Windows\System\Scripts\Shutdown
HKLM\Software\Microsoft\Windows\CurrentVersion\Group Policy\Scripts\Startup
HKLM\Software\Microsoft\Windows\CurrentVersion\Group Policy\Scripts\Shutdown

**Systemwide ASEPs in the registry—64-bit only**

HKLM\Software\Wow6432Node\Microsoft\Windows\CurrentVersion\Run
HKLM\Software\Wow6432Node\Microsoft\Windows\CurrentVersion\RunOnce
HKLM\Software\Wow6432Node\Microsoft\Windows\CurrentVersion\RunOnceEx
HKLM\Software\Wow6432Node\Microsoft\Active Setup\Installed Components

**Systemwide ActiveSync ASEPs in the registry**

HKLM\Software\Microsoft\Windows CE Services\AutoStartOnConnect
HKLM\Software\Microsoft\Windows CE Services\AutoStartOnDisconnect

**Systemwide ActiveSync ASEPs in the registry—64-bit only**

HKLM\Software\Wow6432Node\Microsoft\Windows CE Services\AutoStartOnConnect
HKLM\Software\Wow6432Node\Microsoft\Windows CE Services\AutoStartOnDisconnect

## Explorer

The Explorer tab lists common autostart entries that hook directly into Windows Explorer[3] and usually run in-process with Explorer.exe. Again, although most entries are systemwide, there are a number of per-user entries. Key entries on the Explorer tab include the following:

- Shell extensions that add context menu items, modify property pages, and control column displays in folder windows
- Namespace extensions such as the Desktop, Control Panel, and Recycle Bin, as well as third-party namespace extensions
- Pluggable namespace handlers, which handle standard protocols such as http, ftp, and mailto, as well as Microsoft or third-party extensions such as about, mk, and res
- Pluggable MIME filters

On 64-bit versions of Windows, in-process components such as DLLs can be loaded only into processes built for the same CPU architecture. For example, shell extensions implemented as 32-bit DLLs can be loaded only into the 32-bit version of Windows Explorer—and 64-bit Windows uses the 64-bit Explorer by default. Therefore, these extensions might not appear to work at all on 64-bit Windows.

The following lists the Explorer ASEP locations that Autoruns inspects on a particular instance of an x64 version of Windows 10.

| **Per-user ASEPs under HKCU\Software** |
| --- |
| HKCU\Software\Classes*\ShellEx\ContextMenuHandlers<br>HKCU\Software\Classes*\ShellEx\PropertySheetHandlers<br>HKCU\Software\Classes\AllFileSystemObjects\ShellEx\ContextMenuHandlers<br>HKCU\Software\Classes\AllFileSystemObjects\ShellEx\DragDropHandlers<br>HKCU\Software\Classes\AllFileSystemObjects\ShellEx\PropertySheetHandlers<br>HKCU\Software\Classes\Clsid\{AB8902B4-09CA-4bb6-B78D-A8F59079A8D5}\Inprocserver32<br>HKCU\Software\Classes\Directory\Background\ShellEx\ContextMenuHandlers<br>HKCU\Software\Classes\Directory\ShellEx\ContextMenuHandlers<br>HKCU\Software\Classes\Directory\Shellex\CopyHookHandlers<br>HKCU\Software\Classes\Directory\Shellex\DragDropHandlers<br>HKCU\Software\Classes\Directory\Shellex\PropertySheetHandlers<br>HKCU\Software\Classes\Drive\ShellEx\ContextMenuHandlers<br>HKCU\Software\Classes\Folder\Shellex\ColumnHandlers<br>HKCU\Software\Classes\Folder\ShellEx\ContextMenuHandlers<br>HKCU\Software\Classes\Folder\ShellEx\DragDropHandlers<br>HKCU\Software\Classes\Folder\ShellEx\ExtShellFolderViews<br>HKCU\Software\Classes\Folder\ShellEx\PropertySheetHandlers<br>HKCU\Software\Classes\Protocols\Filter<br>HKCU\Software\Classes\Protocols\Handler<br>HKCU\Software\Microsoft\Ctf\LangBarAddin<br>HKCU\Software\Microsoft\Internet Explorer\Desktop\Components<br>HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\ShellIconOverlayIdentifiers<br>HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\ShellServiceObjects<br>HKCU\Software\Microsoft\Windows\CurrentVersion\ShellServiceObjectDelayLoad |

[3] Windows Explorer was renamed "File Explorer" beginning in Windows 8.

| Systemwide ASEPs in the registry |
|---|
| HKLM\Software\Classes*\ShellEx\ContextMenuHandlers<br>HKLM\Software\Classes*\ShellEx\PropertySheetHandlers<br>HKLM\Software\Classes\AllFileSystemObjects\ShellEx\ContextMenuHandlers<br>HKLM\Software\Classes\AllFileSystemObjects\ShellEx\DragDropHandlers<br>HKLM\Software\Classes\AllFileSystemObjects\ShellEx\PropertySheetHandlers<br>HKLM\Software\Classes\Directory\Background\ShellEx\ContextMenuHandlers<br>HKLM\Software\Classes\Directory\ShellEx\ContextMenuHandlers<br>HKLM\Software\Classes\Directory\Shellex\CopyHookHandlers<br>HKLM\Software\Classes\Directory\Shellex\DragDropHandlers<br>HKLM\Software\Classes\Directory\Shellex\PropertySheetHandlers<br>HKLM\Software\Classes\Drive\ShellEx\ContextMenuHandlers<br>HKLM\Software\Classes\Folder\Shellex\ColumnHandlers<br>HKLM\Software\Classes\Folder\ShellEx\ContextMenuHandlers<br>HKLM\Software\Classes\Folder\ShellEx\DragDropHandlers<br>HKLM\Software\Classes\Folder\ShellEx\ExtShellFolderViews<br>HKLM\Software\Classes\Folder\ShellEx\PropertySheetHandlers<br>HKLM\Software\Classes\Protocols\Filter<br>HKLM\Software\Classes\Protocols\Handler |
| HKLM\Software\Microsoft\Ctf\LangBarAddin<br>HKLM\Software\Microsoft\Windows\CurrentVersion\Explorer\SharedTaskScheduler<br>HKLM\Software\Microsoft\Windows\CurrentVersion\Explorer\ShellExecuteHooks<br>HKLM\Software\Microsoft\Windows\CurrentVersion\Explorer\ShellIconOverlayIdentifiers<br>HKLM\Software\Microsoft\Windows\CurrentVersion\Explorer\ShellServiceObjects<br>HKLM\Software\Microsoft\Windows\CurrentVersion\ShellServiceObjectDelayLoad |

| Systemwide ASEPs in the registry—64-bit only |
|---|
| HKLM\Software\Wow6432Node\Classes*\ShellEx\ContextMenuHandlers<br>HKLM\Software\Wow6432Node\Classes*\ShellEx\PropertySheetHandlers<br>HKLM\Software\Wow6432Node\Classes\AllFileSystemObjects\ShellEx\ContextMenuHandlers<br>HKLM\Software\Wow6432Node\Classes\AllFileSystemObjects\ShellEx\DragDropHandlers<br>HKLM\Software\Wow6432Node\Classes\AllFileSystemObjects\ShellEx\PropertySheetHandlers<br>HKLM\Software\Wow6432Node\Classes\Directory\Background\ShellEx\ContextMenuHandlers<br>HKLM\Software\Wow6432Node\Classes\Directory\ShellEx\ContextMenuHandlers<br>HKLM\Software\Wow6432Node\Classes\Directory\Shellex\CopyHookHandlers<br>HKLM\Software\Wow6432Node\Classes\Directory\Shellex\DragDropHandlers<br>HKLM\Software\Wow6432Node\Classes\Directory\Shellex\PropertySheetHandlers<br>HKLM\Software\Wow6432Node\Classes\Drive\ShellEx\ContextMenuHandlers<br>HKLM\Software\Wow6432Node\Classes\Folder\Shellex\ColumnHandlers<br>HKLM\Software\Wow6432Node\Classes\Folder\ShellEx\ContextMenuHandlers<br>HKLM\Software\Wow6432Node\Classes\Folder\ShellEx\DragDropHandlers<br>HKLM\Software\Wow6432Node\Classes\Folder\ShellEx\ExtShellFolderViews<br>HKLM\Software\Wow6432Node\Classes\Folder\ShellEx\PropertySheetHandlers<br>HKLM\Software\Wow6432Node\Microsoft\Windows\CurrentVersion\Explorer\SharedTaskScheduler<br>HKLM\Software\Wow6432Node\Microsoft\Windows\CurrentVersion\Explorer\ShellExecuteHooks<br>HKLM\Software\Wow6432Node\Microsoft\Windows\CurrentVersion\Explorer\ShellIconOverlayIdentifiers<br>HKLM\Software\Wow6432Node\Microsoft\Windows\CurrentVersion\Explorer\ShellServiceObjects<br>HKLM\Software\Wow6432Node\Microsoft\Windows\CurrentVersion\ShellServiceObjectDelayLoad |

## Internet Explorer

Internet Explorer is designed for extensibility, with interfaces specifically exposed to enable Explorer bars such as the Favorites and History bars, toolbars, and custom menu items and toolbar buttons. And Browser Helper Objects (BHOs) enable almost limitless possibilities for extending the capabilities and user experiences for Internet Explorer.

However, because so much of users' computer time is spent in a browser, and because much of the high-value information that users handle (such as passwords and credit card information) goes through the browser, it has become a primary target of attackers. The same programmatic interfaces that enable integration with third-party document readers and instant messaging have also been used by spyware, adware, and other malicious endeavors.

The following lists the Internet Explorer ASEP locations that Autoruns inspects on a particular instance of an x64 version of Windows 10.

| **Per-user ASEPs under HKCU\Software** |
|---|
| HKCU\Software\Microsoft\Internet Explorer\Explorer Bars<br>HKCU\Software\Microsoft\Internet Explorer\Extensions<br>HKCU\Software\Microsoft\Internet Explorer\UrlSearchHooks |
| **Systemwide ASEPs in the registry** |
| HKLM\Software\Microsoft\Internet Explorer\Explorer Bars<br>HKLM\Software\Microsoft\Internet Explorer\Extensions<br>HKLM\Software\Microsoft\Internet Explorer\Toolbar<br>HKLM\Software\Microsoft\Windows\CurrentVersion\Explorer\Browser Helper Objects |
| **Per-user and systemwide ASEPs in the registry—64-bit only** |
| HKCU\Software\Wow6432Node\Microsoft\Internet Explorer\Explorer Bars<br>HKCU\Software\Wow6432Node\Microsoft\Internet Explorer\Extensions<br>HKLM\Software\Wow6432Node\Microsoft\Internet Explorer\Explorer Bars<br>HKLM\Software\Wow6432Node\Microsoft\Internet Explorer\Extensions<br>HKLM\Software\Wow6432Node\Microsoft\Internet Explorer\Toolbar<br>HKLM\Software\Wow6432Node\Microsoft\Windows\CurrentVersion\Explorer\Browser Helper Objects |

## Scheduled Tasks

The Scheduled Tasks tab displays entries that are configured to be launched by the Windows Task Scheduler. The Task Scheduler allows programs to be launched on a fixed schedule or upon triggering events, such as a user logging on or the computer being idle for a period of time. Commands scheduled with At.exe also appear in the list. The Task Scheduler was greatly enhanced in Windows Vista, so Windows now makes heavy use of it, and the list on the Scheduled Tasks tab will generally be long unless you hide verified Windows entries.

Because tasks can actually be disabled in Windows (unlike Start menu items), clearing the check box next to a scheduled task in Autoruns disables the task rather than copying it to a backup location.[4]

If you select Jump To Entry from the Entry menu for a scheduled task entry, Autoruns displays the Task Scheduler user interface, but it does not try to navigate to the selected entry.

4 "At" jobs cannot be disabled, whether using Autoruns or the Windows Task Scheduler. "At" jobs can be deleted. Note that AT.EXE was deprecated and no longer works on Windows 8 or newer.

## Services

Windows services run in noninteractive, user-mode processes that can be configured to start independently of any user logging on, and that are controlled through a standard interface with the Service Control Manager. Multiple services can be configured to share a single process. A common example of this can be seen in Svchost.exe (Host Process for Windows Services), which is specifically designed to host multiple services implemented in separate DLLs.

Services are configured in the subkeys of HKLM\System\CurrentControlSet\Services. The *Start* value within each subkey determines whether and how the service starts.

Autoruns' Services tab lists services that are not disabled, unless they were disabled by Autoruns (indicated by the presence of an *AutorunsDisabled* value in the service's registry key). The content for the Description column comes from the text or the resource identified by the *Description* value in the configuration key. The image path column displays the path to the service executable; for Svchost services, Autoruns displays the path to the target DLL identified by the *ServiceDll* value in the service's key or its *Parameters* subkey. There are cases for some services in some versions of Windows where administrative rights are required to view the Parameters key; in these cases, Autoruns displays the path to Svchost.exe in the image path column.

Be certain you know what you are doing when disabling or deleting services. Missteps can leave your system with degraded performance, unstable, or unbootable. And again, note that disabling or deleting a service does not stop the service if it is already running.

One malware technique to watch for is a service that looks like it's supposed to be part of Windows but isn't, such as a file named *svchost.exe* in the Windows directory instead of in System32. Another technique is to make legitimate services dependent on a malware service; removing or disabling the service without fixing the dependency can result in an unbootable system. Autoruns' Jump To Entry feature is handy for verifying whether the service's configuration in the registry includes a *DependOnService* value that you can inspect for dependencies before making changes.

## Drivers

Like services, drivers are also configured in the subkeys of HKLM\System\CurrentControlSet\Services, as well as in HKLM\Software\Microsoft\Windows NT\CurrentVersion\Font Drivers. Unlike services, drivers run in kernel mode, thus becoming part of the core of the operating system. Most are installed in System32\Drivers and have a .sys file extension. Drivers enable Windows to interact with various types of hardware, including displays, storage, smartcard readers, and human input devices. They are also used to monitor network traffic and file I/O by antivirus software (and by Sysinternals utilities such as Procmon and Procexp!). And, of course, they are also used by malware, particularly rootkits.

As with services, the Drivers tab displays drivers that are not marked as disabled, except those disabled through Autoruns. The *Description* value comes from the version resource of the driver file, and the image path points to the location of the driver file.

Most blue-screen crashes are caused by an illegal operation performed in kernel mode, and most of those are caused by a bug in a third-party driver. (Less common reasons for blue screens are faulty hardware, the termination of a system-critical process such as Csrss.exe, or an intentional crash triggered through the keyboard driver's crash functionality, as described in Knowledge Base article 244139: *http://support.microsoft.com/kb/244139*.)

You can disable or delete a problematic driver with Autoruns. Doing so will usually take effect after a reboot. As with services, be absolutely certain you know what you are doing when disabling or deleting the configuration of drivers. Many are critical to the operating system, and any misconfiguration might prevent Windows from working at all.

## Codecs

The Codecs category lists executable code that can be loaded by media playback applications. Buggy or misconfigured codecs have been known to cause system slowdowns and other problems, and these ASEPs have also been abused by malware. The following lists the keys that are shown on the Codecs tab.

| **Keys inspected under both HKLM and HKCU** |
|---|
| \Software\Classes\CLSID\{083863F1-70DE-11d0-BD40-00A0C911CE86}\Instance<br>\Software\Classes\CLSID\{7ED96837-96F0-4812-B211-F13C24117ED3}\Instance<br>\Software\Classes\CLSID\{ABE3B9A4-257D-4B97-BD1A-294AF496222E}\Instance<br>\Software\Classes\CLSID\{AC757296-3522-4E11-9862-C17BE5A1767E}\Instance<br>\Software\Classes\Filter<br>\Software\Microsoft\Windows NT\CurrentVersion\Drivers32 |
| **Keys inspected under both HKLM and HKCU on 64-bit Windows** |
| \Software\Wow6432Node\Classes\CLSID\{083863F1-70DE-11d0-BD40-00A0C911CE86}\Instance<br>\Software\Wow6432Node\Classes\CLSID\{7ED96837-96F0-4812-B211-F13C24117ED3}\Instance<br>\Software\Wow6432Node\Classes\CLSID\{ABE3B9A4-257D-4B97-BD1A-294AF496222E}\Instance<br>\Software\Wow6432Node\Classes\CLSID\{AC757296-3522-4E11-9862-C17BE5A1767E}\Instance<br>\Software\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Drivers32 |

## Boot Execute

The Boot Execute tab shows you Windows native-mode executables that are started by the Session Manager (Smss.exe) during system boot. BootExecute typically includes tasks, such as hard-drive verification and repair (Autochk.exe), that cannot be performed while Windows is running. The Execute, S0InitialCommand, and SetupExecute entries should never be populated after Windows has been installed. The following lists the keys that are displayed on the Boot Execute tab.

| Keys that are displayed on the Boot Execute tab |
| --- |
| HKLM\System\CurrentControlSet\Control\ServiceControlManagerExtension<br>HKLM\System\CurrentControlSet\Control\Session Manager\BootExecute<br>HKLM\System\CurrentControlSet\Control\Session Manager\Execute<br>HKLM\System\CurrentControlSet\Control\Session Manager\S0InitialCommand<br>HKLM\System\CurrentControlSet\Control\Session Manager\SetupExecute |

## Image hijacks

*Image hijacks* is the term I use for ASEPs that run a different program from the one you specify and expect to be running. The Image Hijacks tab displays four types of these redirections:

- **exefile** Changes to the association of the .exe or .cmd file types with an executable command. The file-association user interfaces in Windows have never exposed a way to change the association of the .exe or .cmd file types, but they can be changed in the registry. Note that there are per-user and systemwide versions of these ASEPs.
- **htmlfile** Changes to the association of the .htm or .html file types with an executable command. Some malware that hijacks these ASEPs can come into play when you open an HTML file. Verify that the executable command is a legitimate browser.
- **Command Processor\Autorun** A command line that is executed whenever a new Cmd.exe instance is launched. The command runs within the context of the new Cmd.exe instance. There is a per-user and systemwide variant, as well as a separate version for the 32-bit Cmd.exe on 64-bit Windows.
- **Image File Execution Options (IFEO)** Subkeys of this registry location (and its echo in the 64-bit versions of Windows) are used for a number of internal and undocumented purposes. One purpose for IFEO subkeys that *has* been documented is the ability to specify an alternate program to start whenever a particular application is launched. By creating a subkey named for the file name of the original program and a "Debugger" value within that key that specifies an executable path to an alternate program, the alternate program is started instead and receives the original program path and command line on its command line. The original purpose of this mechanism was for the alternate program to be a debugger and for the new process to be started by that debugger, rather than having a debugger attach to the process later, after its startup code had already run. However, there is no requirement that the alternate program actually be a debugger, nor that it even look at the command line passed to it. In fact, this mechanism is how Process Explorer (described in Chapter 3) replaces Task Manager.

The following list shows the registry keys corresponding to these ASEPS that are shown on the Image Hijacks tab.

| **Registry locations inspected for EXE file hijacks** |
|---|
| HKCU\Software\Classes\Exefile\Shell\Open\Command\(Default)<br>HKCU\Software\Classes\.exe<br>HKCU\Software\Classes\.cmd<br>HKLM\Software\Classes\Exefile\Shell\Open\Command\(Default)<br>HKLM\Software\Classes\.exe<br>HKLM\Software\Classes\.cmd |
| **Registry locations inspected for htmlfile hijacks** |
| HKCU\Software\Classes\Htmlfile\Shell\Open\Command\(Default)<br>HKLM\Software\Classes\Htmlfile\Shell\Open\Command\(Default) |
| **Command processor autorun keys** |
| HKCU\Software\Microsoft\Command Processor\Autorun<br>HKLM\Software\Microsoft\Command Processor\Autorun<br>HKLM\Software\Wow6432Node\Microsoft\Command Processor\Autorun |
| **Keys inspected for Image File Execution Options hijacks** |
| HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options<br>HKLM\Software\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Image File Execution Options |

## AppInit

The idea behind AppInit DLLs surely seemed like a good idea to the software engineers who incorporated it into Windows NT 3.1. Specify one or more DLLs in the Appinit_Dlls registry key, and those DLLs will be loaded into every process that loads User32.dll (that is, virtually all user-mode Windows processes). Well, what could go wrong with that?

- The AppInit DLLs are loaded into the process during User32's initialization—that is, while its *DllMain* function is executing. Developers are explicitly told not to load other DLLs within a *DllMain*. It can lead to deadlocks and out-of-order loads, which can lead to application crashes. And yet here, the AppInit DLL "feature" does exactly that. And yes, that has led to deadlock and application crashes.5
- A DLL that automatically gets loaded into every process on the computer sounds like a winner if you are writing malware. Although AppInit has been used in legitimate (but misguided) software, it is frequently used by malware.

Because of these problems, AppInit DLLs are deprecated and disabled by default in Windows Vista and newer. For purposes of backward compatibility, it is possible to re-enable AppInit DLL functionality, but doing so is strongly discouraged. To ensure that AppInit DLLs have not been re-enabled, verify that the *LoadAppInit_DLLs* DWORD value is 0 in *HKLM\Software\Microsoft\Windows NT\CurrentVersion\Windows* and in *HKLM\Software\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Windows*.

[5] Raymond Chen wrote a blog post about AppInit DLLs that he titled "AppInit_DLLs should be renamed Deadlock_Or_Crash_Randomly_DLLs": *https://blogs.msdn.microsoft.com/oldnewthing/20071213-00/?p=24183/*

| Registry values inspected for AppInit Entries |
| --- |
| HKLM\Software\Microsoft\Windows NT\CurrentVersion\Windows\Appinit_Dlls<br>HKLM\Software\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Windows\Appinit_Dlls<br>HKLM\System\CurrentControlSet\Control\Session Manager\AppCertDlls |

## KnownDLLs

KnownDLLs helps improve system performance by ensuring that all Windows processes use the same version of certain DLLs, rather than choose their own from various file locations. During startup, the Session Manager maps the DLLs listed in HKLM\System\CurrentControlSet\Control\Session Manager\KnownDlls into memory as named section objects. When a new process is loaded and needs to map these DLLs, it uses the existing sections rather than searching the file system for another version of the DLL.

The Autoruns KnownDLLs tab should contain only verifiable Windows DLLs. On 64-bit versions of Windows, the KnownDLLs tab lists one ASEP, but file entries are duplicated for both 32-bit and 64-bit versions of the DLLs, in directories specified by the *DllDirectory* and *DllDirectory32* values in the registry key. Note that the Windows-On-Windows-64 (WOW64) support DLLs are present only in the System32 directory and Autoruns will report “file not found” for the corresponding SysWOW64 directory entries. This is normal.

To verify that malware hasn’t deleted an entry from this key so that it can load its own version of a system DLL, save the Autoruns results from the suspect system and compare it against the results from a known-good instance of the same operating system. See the “Saving and comparing results” section later in this chapter for more information.

## Winlogon

The Winlogon tab displays entries that hook into Winlogon.exe, which manages the Windows interactive-logon user interface. Introduced in Windows Vista, the Credential Provider interface manages the user authentication interface. Today, Windows includes many credential providers that handle password, PIN, picture-password, smartcard, and biometric logon. Most of these are shown only if you disable the Hide Windows Entry option. Third parties can supply credential providers that further customize interactive user logons.

The Winlogon tab also includes the user’s configured screen saver, which is started by Winlogon.exe after inactivity, and registered Group Policy client-side extensions (CSEs), which are DLLs that the Group Policy engine loads. The Group Policy engine used to run in the Winlogon process, but now it runs in the Group Policy Client service.

The following list specifies the registry keys that are shown on the Winlogon tab.

| Per-user specification of the screen saver |
|---|
| HKCU\Control Panel\Desktop\Scrnsave.exe |

| Per-user specification of the screen saver, controlled by Group Policy |
|---|
| HKCU\Software\Policies\Microsoft\Windows\Control Panel\Desktop\Scrnsave.exe |

| Group Policy Client-Side Extensions (CSEs) |
|---|
| HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\GPExtensions<br>HKLM\Software\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon\GPExtensions |

| Credential provider ASEPs |
|---|
| HKLM\Software\Microsoft\Windows\CurrentVersion\Authentication\Credential Provider Filters<br>HKLM\Software\Microsoft\Windows\CurrentVersion\Authentication\Credential Providers<br>HKLM\Software\Microsoft\Windows\CurrentVersion\Authentication\PLAP Providers |

| Systemwide identification of a program to verify successful boot |
|---|
| HKLM\System\CurrentControlSet\Control\BootVerificationProgram\ImagePath |

| ASEP for custom setup and deployment tasks |
|---|
| HKLM\System\Setup\CmdLine |

## Winsock providers

Windows Sockets (Winsock) is an extensible API on Windows because third parties can add a *transport service provider* that interfaces Winsock with other protocols or layers on top of existing protocols to provide functionality such as proxying. Third parties can also add a *namespace service provider* to augment Winsock's name-resolution facilities. Service providers plug into Winsock by using the Winsock *service provider interface* (SPI). When a transport service provider is registered with Winsock, Winsock uses the transport service provider to implement socket functions, such as *connect* and *accept*, for the address types that the provider indicates it implements. There are no restrictions on how the transport service provider implements the functions, but the implementation usually involves communicating with a transport driver in kernel mode.

The Winsock tab lists the providers registered on the system, including those that are built into Windows. You can hide the latter group by enabling Hide Windows Entries and Verify Code Signatures to focus on the entries that are more likely to be causing problems.

| Keys inspected for Winsock Provider Entries |
|---|
| HKLM\System\CurrentControlSet\Services\WinSock2\Parameters\NameSpace_Catalog5\Catalog_Entries<br>HKLM\System\CurrentControlSet\Services\WinSock2\Parameters\NameSpace_Catalog5\Catalog_Entries64<br>HKLM\System\CurrentControlSet\Services\WinSock2\Parameters\Protocol_Catalog9\Catalog_Entries<br>HKLM\System\CurrentControlSet\Services\WinSock2\Parameters\Protocol_Catalog9\Catalog_Entries64 |

## Print monitors

The entries listed on the Print Monitors tab are DLLs that are configured in the subkeys of HKLM\System\CurrentControlSet\Control\Print\Monitors. These DLLs are loaded into the Spooler service, which runs as Local System.

> **Note** One of the most common problems that affects the print spooler is misbehaving or poorly coded third-party port monitors. A good first step in troubleshooting print spooler issues is to disable third-party port monitors to see whether the problem persists.

## LSA providers

This category of autostarts comprises packages that define or extend user authentication for Windows, via the Local Security Authority (LSA). Unless you have installed third-party authentication packages or password filters, this list should contain only Windows-verifiable entries. The DLLs listed in these entries are loaded by Lsass.exe or Winlogon.exe and run as Local System.

The SecurityProviders ASEP that is also shown on this tab lists registered cryptographic providers. DLLs listed in this ASEP get loaded into many privileged and standard user processes, so this ASEP has been targeted as a malware persistence vector. (This ASEP isn't truly related to the LSA, except that, like the LSA, it represents security-related functionality.)

| Keys inspected for Authentication Providers |
| --- |
| HKLM\System\CurrentControlSet\Control\Lsa\Authentication Packages<br>HKLM\System\CurrentControlSet\Control\Lsa\Notification Packages<br>HKLM\System\CurrentControlSet\Control\Lsa\Security Packages<br>HKLM\System\CurrentControlSet\Control\Lsa\OSConfig\Security Packages |

| Keys inspected for Registered Cryptographic Providers |
| --- |
| HKLM\System\CurrentControlSet\Control\SecurityProviders\SecurityProviders |

## Network providers

The Network Providers tab lists the installed providers handling network communication, which are configured in HKLM\System\CurrentControlSet\Control\NetworkProvider\Order. On a Windows desktop operating system, for example, this tab includes the default providers that provide access to SMB (file and print) servers, Microsoft RDP (Terminal Services/Remote Desktop) servers, and access to WebDAV servers. Additional providers are often visible in this list if you have a more heterogeneous network or additional types of servers that Windows needs to connect to. All entries in this list should be verifiable.

## WMI

The WMI tab lists registered WMI event consumers that can be configured to run arbitrary scripts or command lines when a particular event occurs. When you select an entry on the WMI tab, the lower panel reports information about the target file, the event consumer's full command line, and the condition, such as a WQL query, that will trigger the event consumer to execute.

When you disable a WMI entry, Autoruns replaces the entry with a clone that has the same name but with "_disabled" appended. This breaks the binding to the event filter so that it won't execute. By re-enabling, the original name and the event binding is reestablished.

These events and bindings are stored in the WMI repository in the *ROOT\subscription* namespace.

## Sidebar gadgets

On Windows Vista and Windows 7, this tab lists the Sidebar Gadgets (called "Desktop Gadgets" on Windows 7) that are configured to appear on the user's desktop. Although gadget software is often (but not always) installed in a systemwide location such as %ProgramFiles%, the configuration of which gadgets to run is in %LOCALAPPDATA%\Microsoft\Windows Sidebar\Settings.ini, which is per-user and nonroaming. Disabling or deleting gadgets with Autoruns manipulates entries in the Settings.ini file.

The image path usually points to an XML file. The gadgets that shipped with Windows Vista and Windows 7 are catalog signed and can be verified. Gadgets were discontinued after Windows 7.

## Office

The Office tab lists add-ins and plug-ins registered to hook into documented interfaces for Access, Excel, Outlook, PowerPoint, and Word. On 64-bit Windows, Office add-ins can be registered to run in 32-bit or 64-bit Office versions. 32-bit add-ins are registered in *Wow6432Node* subkeys on 64-bit Windows.

| **Keys inspected under both HKLM and HKCU** |
| --- |
| \Software\Microsoft\Office\Access\Addins<br>\Software\Microsoft\Office\Excel\Addins<br>\Software\Microsoft\Office\Outlook\Addins<br>\Software\Microsoft\Office\PowerPoint\Addins<br>\Software\Microsoft\Office\Word\Addins |
| **Keys inspected under both HKLM and HKCU on 64-bit Windows** |
| \Software\Wow6432Node\Microsoft\Office\Access\Addins<br>\Software\Wow6432Node\Microsoft\Office\Excel\Addins<br>\Software\Wow6432Node\Microsoft\Office\Outlook\Addins<br>\Software\Wow6432Node\Microsoft\Office\PowerPoint\Addins<br>\Software\Wow6432Node\Microsoft\Office\Word\Addins |

## Saving and comparing results

Autoruns results can be saved to disk in two different file formats: tab-delimited text, or a binary format that preserves all the data captured. The binary format can be loaded into Autoruns for viewing at a later time or on a different system, and it can be compared against another set of Autoruns results.

In both cases, the results are read-only: they can't be used to roll back a system to an earlier state or configuration, and after they have been captured, you cannot add or remove options to modify the saved results. You can apply or remove the filters described in the "Hiding entries" section earlier in this chapter to control which entries Autoruns displays.

### Saving as tab-delimited text

Click the Save button on the toolbar; in the Save dialog box, change the Save As Type to *Text (*.txt)*, and specify a file in which to save the current results. The data displayed on the Everything tab is written to the file in five-column or six-column tab-delimited format, depending on whether the Check VirusTotal.com option is enabled. The rows identifying the ASEP locations (the gray-shaded rows in the Autoruns display) include the location in the first column, the location's last-modification timestamp in the fifth column, and empty strings in the remaining columns. The rows identifying Autorun Entries that are enabled (the check boxes are selected) are written to the file prepended with a plus sign (+); those that are disabled are prepended with an X.

The text file can be imported into Microsoft Office Excel. You should specify the first column as Text instead of General so that the leading plus signs do not get interpreted as an instruction or other special character.

The tab-delimited format respects the selections on the Options menu. If Hide Empty Locations is not enabled, the file will include all ASEPs, including those that have no entries. If Hide Microsoft Entries, Hide Windows Entries, or Hide VirusTotal Clean Entries is selected, those entries will be omitted from the output. If Verify Code Signatures is selected, the Publisher column will include Verified or Not Verified, as appropriate. If Check VirusTotal.com is selected, the output adds a sixth column with the VirusTotal column's results.

Note that Autoruns results saved in text format cannot be read back in to Autoruns.

See the section on AutorunsC later in this chapter for a scriptable way to capture Autoruns data to other text file formats.

### Saving in binary (.arn) format

The Autoruns binary file format with its default .arn file extension is the Autoruns "native" file format.Click the Save icon on the toolbar, and specify a file in which to save the results, leaving the Save As Type option as Autoruns Data (*.arn). All information captured in the most recent scan is preserved, including signature verification and VirusTotal results, even for entries that are filtered from the display.

You can automate the capture of Autoruns data and save it to a .arn file with the –**a** command-line option. The following command captures the state of autostart entries on the system to outputfile.arn, using default Autoruns options:

```
Autoruns -a outputfile.arn
```

To add signature verification, include the **–v** option as shown in the following example. Make sure not to put it *between* the –**a** and the file name: the file name must immediately follow the **–a** parameter.

```
Autoruns -v -a outputfile.arn
```

### Viewing and comparing saved results

To view the .arn file on the same or another system, choose Open from the File menu and select the saved file. When Autoruns starts, it creates a file association for .arn, so you can also open a .arn file simply by double-clicking it in Explorer. You can also open a saved file from the Autoruns command line by specifying the file path without any other switches:

```
Autoruns C:\Users\Mark\Desktop\outputfile.arn
```

To compare the results displayed in Autoruns—whether it's a fresh capture or from a saved file—choose Compare from the File menu and select the saved file to compare the displayed results against. Autoruns shows only the entries that have changed between the two sets, with the ones that are present only in the original set highlighted in green, and entries that are only in the "compare" file highlighted in red. Because the content of the Publisher column depends on whether signature verification is enabled, you should compare only captures that have the same signature verification selection.

## AutorunsC

AutorunsC is a console-mode version of Autoruns that outputs results to its standard output. It is designed primarily for use in scripts. Its purpose is data collection only: it cannot disable or delete any autostart entries.

The command-line options are listed in Table 4-1.[6] They let you capture all autostarts or just specific categories, verify digital signatures, query VirusTotal, omit Microsoft entries, specify a user account for which to capture autostarts or capture all user accounts' autostarts, and output results as comma-separated or tab-separated values (CSV) or as XML. If you don't specify any options, AutorunsC outputs just the Logon entries without signature verification and in an indented list format designed for human reading. To capture other ASEPs, add the **–a** option followed by one or more letters indicating the ASEP categories of interest, or * to capture all ASEP categories.

[6] Note that AutorunsC's command-line syntax was completely overhauled in version 13.0, which was released in January, 2015. If you have scripts designed for earlier versions of AutorunsC, you should review and update them.

Whether in the default list format, CSV, or XML, AutorunsC's output always includes the ASEP location, entry name, description, version, publisher, image path, command line, whether the entry is disabled, and the date and time the target file was last modified, according to the file system. CSV output also includes a row for each ASEP location and when it was last modified. Note that because Windows tracks the last write time for registry keys but not for individual registry values, the "last modified" time for a registry ASEP location will be for the key and might not reflect when a specific entry was changed. When signature verification is enabled, CSV output includes both the signing name as well as the Company Name attribute from the file's version resource.

When file hashes are requested with the **–h** option, AutorunsC outputs MD5, SHA-1, SHA-256, and IMPHASH[7] hashes of the target file, as well as PESHA-1 and PESHA-256 hashes that are used for Authenticode signatures and that cover only the content areas and not the filler of Portable Executable (PE) files.

CSV and XML output also explicitly name the user profile to which each entry belongs, or "System-wide" for entries that apply to the entire system.

The CSV format includes column headers, and it imports easily into Excel or relational databases. The XML format is easily consumed by Windows PowerShell or any other XML consumer. For example, the following lines of PowerShell run AutorunsC, read the XML, and then display disabled items:

```
$arcx = [xml]$(autorunsc -a * -x -accepteula)

$arcx.SelectNodes("/autoruns/item") | ?{ $_.enabled -ne "Enabled" }
```

**TABLE 4-1** AutorunsC command-line options

| **Autostart types:** `[-a *|bcdeghiklmoprsw]` | |
|---|---|
| ***** | Shows all autostart entries |
| **b** | Shows boot execute entries |
| **c** | Shows codecs |
| **d** | Shows AppInit DLLs |
| **e** | Shows Explorer add-ons |
| **g** | Shows Sidebar gadgets (Windows Vista and Windows 7) |
| **h** | Shows image hijacks |
| **i** | Shows Internet Explorer add-ons |
| **k** | Shows known DLLs |
| **l** | Shows logon autostart entries (this is the default) |
| **m** | Shows WMI entries |

7 "Import hashing," or IMPHASH, is based on the content and order of a module's import tables, which lists the names of libraries and the APIs used by the module. It is designed to identify related malware samples, and it is described in more detail in *https://www.mandiant.com/blog/tracking-malware-import-hashing/*. VirusTotal discusses their adoption of imphash in *http://blog.virustotal.com/2014/02/virustotal-imphash.html*.

| **Autostart types:** `[-a *|bcdeghiklmoprsw]` | |
|---|---|
| **n** | Shows Winsock protocol and network providers |
| **o** | Shows Office addins |
| **p** | Shows printer monitor DLLs |
| **r** | Shows LSA security providers |
| **s** | Shows services and non-disabled drivers |
| **t** | Shows scheduled tasks |
| **w** | Shows Winlogon entries |
| **What to scan** | |
| ***user*** | Specifies the name of the user account for which autostart entries will be shown. Use DOMAIN\User format for domain accounts. Specify * to scan all user profiles. This option requires administrative rights. |
| ***–z systemroot userprofile*** | Scans an offline Windows system, specifying the file-system paths to the target system's Windows directory and to the target user-profile directory. |
| **File information** | |
| **–h** | Shows file hashes |
| **–s** | Verifies digital signatures |
| **–u** | If VirusTotal check is enabled, **–u** shows only files that are unknown by VirusTotal or that have non-zero detection.<br>If VirusTotal check is not enabled, **–u** shows only unsigned files. |
| **–v[rs]** | Queries VirusTotal for malware based on file hashes.<br>With "r" added, it opens the web browser to VirusTotal report for files with non-zero detection.<br>With "s" added, it uploads files that report as "unknown"—that is, not previously scanned by VirusTotal.<br>(Also, note the meaning of **–u** when used with the **–v[rs]** option.) |
| **–vt** | Accepts the VirusTotal terms of service (TOS) without opening the TOS webpage. |
| **Output format** | |
| **–c** | Prints output as comma-separated values (CSV) |
| **–ct** | Prints output as tab-delimited values |
| **–x** | Prints output as XML |
| **–m** | Hides Microsoft entries. If used with **–s**, hides signed Microsoft entries. |
| **–t** | Shows timestamps in normalized UTC: YYYYMMDD-hhmmss. Alphabetically sorting normalized UTC also produces a chronological sort. |

## Autoruns and malware

One of the goals of most malware is to remain active on an infected system indefinitely. Malware has therefore always used ASEPs. Years ago, it usually just targeted simple locations such as the Run key under HKLM. As malware has become more sophisticated and difficult to identify, its use of ASEPs has become more sophisticated as well. Malware has been implemented as Winsock providers and as print monitors. Not only are such ASEP locations more obscure, but the malware doesn't show up in a process list because it loads as a DLL in an existing, legitimate process. Malware has also become more adept at infecting and running without requiring administrative privileges, because there are increasing numbers of users who only ever have standard user privileges.

In addition, malware often leverages rootkits, which subvert the integrity of the operating system. Rootkits intercept and modify system calls, lying to software that uses documented system interfaces about the state of the system. Rootkits can hide the presence of registry keys and values, files and directories, processes, sockets, user accounts, and more, or they can make software believe something exists when it doesn't. In short, a computer on which malware has run with administrative privileges cannot be trusted to report its own state accurately. Therefore, Autoruns cannot always be expected to identify malicious autostart entries on a system.

That said, not all malware is that sophisticated, and there are still some telltale signs that can point to malware:

- Entries with a well-known publisher such as Microsoft that fail signature verification. (Unfortunately, not all software published by Microsoft is signed.)
- Entries with an image path pointing to a DLL or EXE file that is missing Description or Publisher information (unless the target file is not found).
- A common Windows component that is launched from an unusual or nonstandard location—for example, svchost.exe or another service launching from C:\Windows or C:\Windows\SysWOW64 (instead of from System32) or from C:\System Volume Information.
- Entries with names that can be mistaken for common Windows components, such as those with slight misspellings—for example, "Isass.exe" with a capital "I" instead of a lower-case "L", "scvhost.exe" instead of "svchost.exe," or "iexplorer.exe" with the extra "r" at the end.
- Entries for which the file date and time of the launched program correspond to when problems were first noticed or a breach is discovered to have occurred.
- Disabling or deleting an entry, pressing F5 to refresh the display, and finding the entry still present and enabled. Malware will often monitor its ASEPs and put them back if they get removed.

Malware and antimalware remains a moving target. Today's "best practices" will seem naïve and insufficient tomorrow.

There are some entries you might come across that seem suspicious but are innocuous:

- A default installation of Windows Vista might have a small number of “File not found” entries on the Drivers tab for NetWare IPX drivers and for “IP in IP Tunnel Driver.”
- Default installations of Windows Vista, Windows 7, Windows Server 2008, and Windows Server 2008 R2 might have a WMI entry named “BVTConsumer”. This code is inoperative and can be safely ignored.
- A default installation of Windows 7 might have a small number of entries on the Scheduled Tasks tab under “\Microsoft\Windows” that show an entry name but no further information.
- As explained in the “KnownDLLs” section earlier in this chapter, on 64-bit Windows Autoruns reports “File not found” for WOW64 support DLLs in the SysWOW64 directory. These known DLLs exist only in the System32 directory.

# Index

## A

# B

## C

# D

# E

# F

# L



# M

## N



## O

# P

# Q



# R

# S

# U



# V

# W



# X



# Z